IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

NIGHTINGALE   Atty. Ref.: 550-510

Serial No. 10/764,495   Group: 2128

Filed: January 27, 2004   Examiner: Alhija, Saif A.

For: APPARATUS AND METHOD FOR PERFORMING HARDWARE
       AND SOFTWARE CO-VERIFICATION TESTING

Before the Board of Patent Appeals and Interferences

BRIEF FOR APPELLANT
On Appeal From Final Rejection
From Group Art Unit 2128

John R. Lastova
NIXON & VANDERHYE P.C.
11th Floor, 901 North Glebe Road
Arlington, Virginia 22203-1808
(703) 816-4025
Attorney for Appellants
Nightingale and ARM Limited

1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of

NIGHTINGALE   Atty. Ref.: 550-510

Serial No. 10/764,495   Group: 2128

Filed: January 27, 2004   Examiner: Alhija, Saif A.

For: APPARATUS AND METHOD FOR PERFORMING HARDWARE
        AND SOFTWARE CO-VERIFICATION TESTING

* *

May 30, 2007

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, ARM Limited, a United Kingdom

corporation.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals related to this subject application. There are no

interferences related to this subject application.

III. STATUS OF CLAIMS

Claims 1-51 are pending, rejected, and on appeal.

IV. STATUS OF AMENDMENTS

No amendment was filed after final. A pre-appeal was filed on March 30, 2007.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The ability to effectively and efficiently test and/or validate designs is very important. Typical data processing system designs are rapidly increasing in complexity and often include circuit blocks designed by a variety of different sources or companies. So-called System-on-Chip (SoC) designs that integrate a large number of functional elements on a single integrated circuit have strong advantages in terms of cost and performance, but require significant amounts of validation and testing before the designs can be reliably released for manufacture. This validation and testing requirement is becoming a bottleneck in getting new systems into the market place. Consequently, measures that can improve the efficiency and effectiveness of such validation and testing of designs are strongly advantageous.

One method for hardware integration performance testing is shown in Figure 1A and includes a verification environment 10 formed of a hardware simulation 140, a plurality of signal interface controllers 35, 40, 45, 120, 125, 130, and a test manager 20. But it does not provide a mechanism for coordinating a software based scenario running on a processor. This problem can be illustrated with reference to Figure 1B, which shows the verification environment of Figure

1A, but with the addition of a representation of a processor 150 provided within the hardware simulation 140. Software to be executed by the processor can be provided as user test code 190, which can be compiled and linked to load that software into the ROM 185. The processor can then execute the software by retrieving instructions from the ROM 185 via the bus matrix 70 and the memory controller 165 using the buses 155, 160, and 175. Further, a RAM 180 accessible via the bus 170 can be used for storing data used by the processor when executing the instructions.

As can be seen from Figure 1B, stimulus signals and/or response signals can be transferred to and from the hardware simulation 140 while the processor 150 is executing software stored within the ROM 185, but there is no co-ordination between the software routines executing on the processor 150 and the hardware verification testing being performed under the control of the test manager 20. In other words, the techniques described above with reference to Figures 1A and 1B cannot readily be used to perform co-verification of hardware and software.

The claims relate to performing hardware and software co-verification testing as part of verifying the design of a data processing system. A debugger signal interface controller (e.g., 300 in Figure 3) interfaces with a debugger (e.g., 310) to control the operation of a processing unit associated with the system under verification. A test manager (e.g., 50) sends test controlling messages to the

debugger signal interface controller, and to other signal interface controllers (e.g., 35, 20, 35, 120, 125, and 130) coupled to the system under verification (e.g., 140), identifying test actions to be performed by those interface controllers. The debugger signal interface controller enables the test manager to cause the debugger signal interface controller to perform one or more test actions such that one or more stimulus signals and one or more response signals are passed between the debugger and the debugger signal interface controller during performance of the sequence of verification tests. This allows the test manager to control the operation of the processing unit via the debugger signal interface controller and the debugger in order to coordinate the execution of the software routines with the sequence of verification tests, e.g., hardware stimulus to the system under verification.

The following is a mapping of independent claim 1 onto an example, non-limiting embodiment in the specification.[1]

| Apparatus for performing a sequence of verification tests to perform hardware and software co-verification on a system under verification, comprising: | See Figure 3 which illustrates an apparatus for performing hardware (140) and software (190) co-verification. The hardware simulation 140 represents at least a portion of a System-on-Chip on an integrated circuit modelled as |

---

[1] This mapping in no way limits the claim scope and is not intended to be used in construing the meaning of claim terms.

	Register Transfer Language (RTL).
a plurality of signal interface controllers operable to be coupled to said system under verification, each signal interface controller being operable to perform one or more test actions transferring at least one of one or more stimulus signals and one or more response signals between a corresponding portion of the system under verification and said signal interface controller during performance of said sequence of verification tests;	Signal interface controllers, also referred to as eXtensible Verification Components (XVCs), 35, 40, 45, 120, 125, and 130 are coupled to the system under verification via corresponding buses 55, 60, 65, 105, 110, 115. See Figure 3. Master XVCs 35, 40, 45 have simulation actions to either drive directed FRBM (File Reader BUS Master) format vectors 50 onto the associated bus, or to generate random patterns of transfers constrained by a user configuration file. Page 2, lines 22-25.
a debugger operable to control operation of a processing unit associated with the system under verification, the processing unit being operable to execute software routines;	Debugger 310 shown in Figure 3 controls operation of the processor 150 in Figure 3 and 330 in Figure 4 so that software routines executed by the processor can be coordinated with the sequence of verification tests performed on the hardware simulation 140. Page 14, lines 19-22.
a debugger signal interface controller operable to interface with the debugger and	A debugger signal interface controller 300 is shown in Figure 3.

to perform one or more test actions transferring at least one of one or more stimulus signals and one or more response signals between the debugger and said debugger signal interface controller during performance of said sequence of verification tests; and	When performing the user-defined test scenarios 25, the debugger signal interface controller 300 receives test controlling messages and performs one or more test actions. Page 14, lines 16-19; page 15, line 11-page 16, line 22.
a test manager coupled to said plurality of signal interface controllers and the debugger signal interface controller and operable to transfer test controlling messages to said plurality of signal interface controllers and the debugger signal interface controller identifying the test actions to be performed;	A test manager 20 shown in Figure 3 sends test controlling messages to the XVCs 35, 40, 45, 120, 125, 130 and to the debugger signal interface controller 300 in order to cause it to perform one or more test actions. Page 14, lines 16-19; page 15, line 11-page 16, line 22.
the test manager being operable to control the operation of the processing unit via the debugger signal interface controller and the debugger in order to co-ordinate the execution of said software routines with the sequence of verification tests.	The test manager controls via the debugger 310 the operation of the processor 150 so that software routines executed by the processor can be co-ordinated with the sequence of verification tests performed on the hardware simulation 140. Page 14, lines 19-22; page 15, line 11-page 16, line 22.

Because independent method claim 18 is a method analog of apparatus claim 1 and independent computer product claim 35 is a computer product analog of apparatus claim 1, the mapping provided for claim 1 also applies.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The rejections on appeal include the rejection of claims 1-51 under 35 U.S.C. §101 as being directed to non-statutory subject matter and the rejection of claims 1-51 under 35 U.S.C. §102 as being anticipated by US 6,678,645 Rajsuman.

VII. ARGUMENT

A.   The Claims Recite Statutory Subject Matter

The Examiner fails to demonstrate or explain how any of the rejected claims does not provide a useful, concrete, and tangible result or that it "consists solely of the manipulation of an abstract idea [that] is not concrete or tangibles [sic]." Nor is it understood how the claims can be dismissed as "software per se." The burden of proof for making and supporting rejections is on the Patent Office. See *In re Piasecki*, 223 USPQ 785, 788 (Fed. Cir. 1984).

Contrary to the Examiner's contentions, the appealed claims accomplish a practical application and recite statutory subject matter that provides a useful, concrete, and tangible result. The summary section above and the background

section of the specification describe the real world problems faced with testing

System-on-Chips (SoCs). The summary section in the specification outlines some

of the practical benefits provided by the claimed technology.

Claim 1 recites an "apparatus for performing a sequence of verification tests

to perform hardware and software co-verification on a system under verification."

Clearly, this is a useful apparatus, and the "apparatus" statutory category is well-

recognized as one of the four classes of statutory inventions. The recited apparatus

elements include signal interface controllers coupled to the system under

verification, a processing unit that executes software routines, a debugger that

controls the processing unit, a debugger signal interface controller that causes

stimulus and response signals to be exchanged between the debugger and the

debugger signal interface controller, and a test manager that accomplish the hardware

and software co-verification on a system under verification. The physical structure

recited in apparatus claim 1 that controls a processing unit to execute software

instructions, exchanges physical stimulus and response signals between two of the

elements, and coordinates execution of software routines with a sequence of

hardware verification tests defines statutory subject matter.

Method claim 18 describes steps of physically testing a system using "signal

interface controllers" and "controlling via a debugger execution of software routines

by a processing unit associated with the system under verification." Physical

stimulus signals and response signals are communicated between the controllers and

the system under test. The test manager controls the operation of the processing

unit via the debugger signal interface controller and the debugger in order to co-

ordinate the execution of said software routines with the sequence of verification

tests. The physical operations recited in claim 18 performed by a physical

processing unit using physical signals on a physical system being tested provide a

very tangible and useful result and can not reasonably be understood as "software per

se" as alleged by the Examiner.

Claim 35 defines "a computer program product embodied on a computer-

readable medium for use in performing a sequence of verification tests to perform

hardware and software co-verification on a system under verification." Each

program code element specifically includes language explaining that the "code, when

executed, causes the computer to" perform a specific operation. The Examiner's

dismissal of the entire claim as non-statutory simply because the phrase "for use" in

the preamble is simply inappropriate. The claimed computer program code is

embodied on a computer-readable medium, which is a physical structure. The claim

specifically recites that when executed, that program code causes a computer to

perform the recited tasks for performing a sequence of verification tests to perform

coordinated hardware and software testing.

The apparatus, method, and product claims are tangible, solve a very real

technical problem, and significantly improve the efficiency of designing a data

processing system such as a System-on-Chip (SoC). The signal interface

controller, debugger, debugger signal interface controller, and test manager in claim 1, as well as the steps in method claim 18, may be implemented in a suitably programmed computer or other suitably configured electronic circuitry, all of which is tangible hardware. The computer product claim 26 is directed to a "computer program product embodied on a computer-readable medium for use in performing a sequence of verification tests to perform hardware and software co-verification on a system under verification." The computer-readable medium is tangible hardware. Each program code element in the body of claim 26 specifically recites that "when executed, [the program code element] causes a computer to perform" a specific action, operation, or task. The §101 rejection is improper and should be withdrawn.

B.   The Legal Requirements For Anticipation

To establish that a claim is anticipated, the Examiner must point out where each and every limitation in the claim is found in a single prior art reference. *Scripps Clinic & Research Found. v. Genentec, Inc.,* 927 F.2d 1565 (Fed. Cir. 1991). Every limitation contained in the claims must be present in the reference, and if even one limitation is missing from the reference, then it does not anticipate the claim. *Kloster Speedsteel AB v. Crucible, Inc.,* 793 F.2d 1565 (Fed. Cir. 1986). Rajsuman fails to satisfy this rigorous standard.

C. Rajsuman Lacks A Debugger For Driving A Processing Unit In A System Under Verification To Co-Ordinate Execution Of Software Routines With Verification Tests

Claim 1 recites "a debugger operable to control operation of a processing unit associated with the system under verification, the processing unit being operable to execute software routines."[2] When performing a sequence of verification tests with respect to the system under verification, the test manager issues test control messages to signal interface controllers coupled to that system and to a debugger signal interface controller. The test controlling messages cause the debugger signal interface controller to perform one or more test actions during which stimulus signals and response signals are passed between the debugger and the debugger signal interface controller during performance of the sequence of verification tests. This allows the test manager to *coordinate* the execution of the *software* routines by the processing unit *along with* the *sequence of verification tests*. All the features of claim 1 are not taught by Rajsuman.

In referring to col. 5, lines 43 to 44, the Examiner apparently equates the verification units (VUs) of Figure 5 in Rajsuman with the claimed signal interface controllers. Figure 5 shows that each verification unit is associated with a separate design validation station (DVS) which together represent a SoC having a plurality

---

[2] Analogous language is found in independent method claim 18 and independent product claim 35.

of functional cores. See col.1, lines 6 to 8. The Examiner refers to column 11, lines 53 to 63 of Rajsuman as allegedly teaching the claimed debugger. Appellant disagrees.

This text in column 11 describes timing verification of critical paths of the SoC design. If the test results obtained using the vectors in the testbench data deviate from the simulation timing results, then there is an error that needs debugging. So the text in column 11, lines 53 to 63 simply describes standard debugging activity. Rajsuman fails to disclose using a debugger to <u>control</u> <u>operation</u> of a processing unit included in the hardware being tested which is also executing software routines as part of a software verification test.

On page 2 of the final action, the Examiner also refers to claim 5 and col. 9, lines 16-24 in Rajsuman with regard to the claimed debugger. Both references explain that Rajsuman's system can produce test patterns for "debugging of a fault in the cores of the SoC." See claim 5. But the claimed debugger functionality is different from such general debugging activity. Rather, the claimed debugger controls the processing unit executing one or more software routines to facilitate hardware and software co-verification, which has traditionally proved to be difficult, as explained above and in the background section of the application. The claimed debugger synchronizes the hardware and software activities to achieve hardware and software co-verification of the system under test. The Examiner never identifies where a debugger in Rajsuman controls a processing unit

executing one or more software routines to facilitate hardware and software co-verification.

D. Rajsuman Lacks A Debugger Signal Interface Controller For Performing Test Actions Transferring Stimulus Signals And Response Signals With A Debugger During Verification Tests

The Examiner refers to the same sections of Rajsuman for the claimed debugger signal interface controller. But those same sections do not disclose "debugger signal interface controller operable to interface with the debugger and to perform one or more test actions transferring at least one of one or more stimulus signals and one or more response signals between the debugger and said debugger signal interface controller during performance of said sequence of verification tests."[3] The debugger signal interface controller receives test controlling messages from the test manager which identify the test actions to be performed. In response, the debugger signal interface controller performs those test actions to cause stimulus signals and response signals to pass between the debugger signal interface controller and the debugger. Those stimulus and response signals affect what software routine is run by the processing unit because the debugger controls operation of the processing unit. In this way, the test manager can "control the operation of the processing unit via the debugger signal interface controller and the debugger in order to co-ordinate the execution of said

software routines with the sequence of verification tests." None of this is taught by Rajsuman.

As explained above, the general reference in Rajsuman to debugging does disclose the claimed functionality quoted above. That claimed functionality is not directed to detecting faults in the claimed processing unit, but instead to synchronizing hardware and software activities, thereby enabling a sequence of verification tests to be performed to test for the correct operation of the software and hardware of the system in combination. Rajsuman lacks the coordinated execution of the claimed software routines along with the sequence of hardware verification tests made possible with the claimed debugger and debugger interface controller.

VIII. CONCLUSION

The Examiner fails to demonstrate why the claims are non-statutory. After reading the independent claims as a whole and in light of the real world applications of the claimed technology described in the specification, the conclusion that must be reached is that the claims recite statutory subject matter. The anticipation rejection is also in error because Rajsuman lacks multiple claim features from the independent claims. The absence of even one claim feature from

---

[3] Quoted from independent apparatus claim 1. Analogous language is found in independent method claim 18 and independent product claim 35.
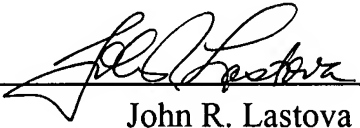
a prior art reference defeats an anticipation rejection. The Board should reverse

the rejections and order the application allowed.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: _____
John R. Lastova
JRL/maa   Reg. No. 33,149
Appendix A - Claims on Appeal

OIPE
MAY 30 2007

IX. CLAIMS APPENDIX

1.   (Original) Apparatus for performing a sequence of verification tests to perform hardware and software co-verification on a system under verification, comprising:

a plurality of signal interface controllers operable to be coupled to said system under verification, each signal interface controller being operable to perform one or more test actions transferring at least one of one or more stimulus signals and one or more response signals between a corresponding portion of the system under verification and said signal interface controller during performance of said sequence of verification tests;

a debugger operable to control operation of a processing unit associated with the system under verification, the processing unit being operable to execute software routines;

a debugger signal interface controller operable to interface with the debugger and to perform one or more test actions transferring at least one of one or more stimulus signals and one or more response signals between the debugger and said debugger signal interface controller during performance of said sequence of verification tests; and

a test manager coupled to said plurality of signal interface controllers and the debugger signal interface controller and operable to transfer test controlling messages

to said plurality of signal interface controllers and the debugger signal interface controller identifying the test actions to be performed;

the test manager being operable to control the operation of the processing unit via the debugger signal interface controller and the debugger in order to co-ordinate the execution of said software routines with the sequence of verification tests.

2.   (Original) Apparatus as claimed in Claim 1, further comprising a memory in which the software routines are stored, the debugger signal interface controller being provided with an address indication identifying the addresses of the software routines within the memory, upon receipt of a test controlling message from the test manager, the debugger signal interface controller being operable to generate a corresponding test action with reference to the address indication.

3.   (Original) Apparatus as claimed in Claim 2, further comprising a status memory operable to store status data, the processing unit being operable to execute monitoring code to monitor the status data in order to identify any changes to the status data and to then execute at least one of said software routines as identified by the change in status data, the corresponding test action being arranged to cause updated status data identifying a corresponding one of said software routines to be stored in the status memory under the control of the debugger.

4.   (Original) Apparatus as claimed in Claim 3, wherein the debugger is operable to cause the processing unit to store the updated status data in the status memory, whereafter the processing unit reverts to executing the monitoring code.

5.   (Original) Apparatus as claimed in Claim 2, wherein the processing unit has a plurality of registers associated therewith, and the corresponding test action is arranged to cause data in a selected register of said plurality of registers to be updated under the control of the debugger, such that the processing unit will then execute a corresponding one of said software routines.

6.   (Original) Apparatus as claimed in Claim 5, wherein the selected register is operable to store a program counter value and the corresponding test action is arranged to cause the program counter value to be updated in order to cause the processing unit to branch to the corresponding one of said software routines.

7.   (Original) Apparatus as claimed in Claim 1, wherein upon execution of one of said software routines, the processing unit is operable to update status data indicative of whether the software routine completed successfully, the debugger signal interface controller being operable to perform a predetermined test action in order to cause a breakpoint to be set by the debugger which is triggered when the processing unit performs said update of the status data.

8.   (Original) Apparatus as claimed in Claim 7, wherein the debugger is operable to issue a callback event to the debugger signal interface controller upon triggering of the breakpoint.

9.   (Original) Apparatus as claimed in Claim 1, wherein said stimulus and/or response signals are transferred between the debugger signal interface controller and the debugger using Application Programming Interface (API) calls.

10.   (Original) Apparatus as claimed in Claim 2, wherein at least one of the software routines is written into the memory via the debugger under the control of the debugger signal interface controller.

11.   (Original) Apparatus as claimed in Claim 1, wherein multiple processing units are provided, and a corresponding multiple of debugger signal interface controllers are provided, each debugger signal interface controller communicating with the same debugger to cause their respective test actions to be performed.

12.   (Original) Apparatus as claimed in Claim 1, wherein the timing of the execution of said software routines is co-ordinated with the sequence of verification tests.

13.   (Original) Apparatus as claimed in Claim 1, wherein the system under verification comprises a plurality of components, each signal interface controller being associated with one of said components.

14.   (Original) Apparatus as claimed in Claim 13, wherein the processing unit forms one of the components of the system under verification.

15.   (Original) Apparatus as claimed in Claim 1, further comprising the processing unit, the processing unit being provided externally to the system under verification.

16.   (Original) Apparatus as claimed in Claim 1, wherein the processing unit comprises a representation of a processor on which the software routines are intended to be executed.

17.   (Original) Apparatus as claimed in Claim 1, wherein the system under verification comprises a hardware simulator responsive to said one or more stimulus signals to generate said one or more response signals simulating a response of a data processing apparatus to said one or more stimulus signals if applied to said data processing apparatus.

18.   (Original) A method of performing a sequence of verification tests to perform hardware and software co-verification on a system under verification, comprising the steps of:

performing in each of a plurality of signal interface controllers coupled to said system under verification one or more test actions transferring at least one of one or more stimulus signals and one or more response signals between a corresponding portion of the system under verification and said signal interface controller during performance of said sequence of verification tests;

controlling via a debugger execution of software routines by a processing unit associated with the system under verification;

performing in a debugger signal interface controller coupled with the debugger one or more test actions transferring at least one of one or more stimulus signals and one or more response signals between the debugger and said debugger signal interface controller during performance of said sequence of verification tests; and

transferring test controlling messages from a test manager to said plurality of signal interface controllers and to the debugger signal interface controller, the test controlling messages identifying the test actions to be performed;

whereby the test manager controls the operation of the processing unit via the debugger signal interface controller and the debugger in order to co-ordinate the execution of said software routines with the sequence of verification tests.

19. (Original) A method as claimed in Claim 18, further comprising the steps of:

storing the software routines in a memory;

providing the debugger signal interface controller with an address indication identifying the addresses of the software routines within the memory; and

upon receipt of a test controlling message from the test manager, generating in the debugger signal interface controller a corresponding test action with reference to the address indication.

20. (Original) A method as claimed in Claim 19, further comprising the steps of:

storing status data in a status memory;

executing on the processing unit monitoring code to monitor the status data in order to identify any changes to the status data and then executing at least one of said software routines as identified by the change in status data, the corresponding test action causing updated status data identifying a corresponding one of said software routines to be stored in the status memory under the control of the debugger.

21. (Original) A method as claimed in Claim 20, further comprising the step of:

causing the processing unit to store the updated status data in the status memory, whereafter the processing unit reverts to executing the monitoring code.

22.   (Original) A method as claimed in Claim 19, wherein the processing unit has a plurality of registers associated therewith, and the corresponding test action causes data in a selected register of said plurality of registers to be updated under the control of the debugger, such that the processing unit will then execute a corresponding one of said software routines.

23.   (Original) A method as claimed in Claim 22, wherein the selected register is operable to store a program counter value and the corresponding test action causes the program counter value to be updated in order to cause the processing unit to branch to the corresponding one of said software routines.

24.   (Original) A method as claimed in Claim 18, further comprising the steps of:

upon execution of one of said software routines, employing the processing unit to update status data indicative of whether the software routine completed successfully; and

causing the debugger signal interface controller to perform a predetermined test action in order to cause a breakpoint to be set by the debugger which is triggered when the processing unit performs said update of the status data.

25.   (Original) A method as claimed in Claim 24, wherein the debugger issues a callback event to the debugger signal interface controller upon triggering of the breakpoint.

26.   (Original) A method as claimed in Claim 18, wherein said stimulus and/or response signals are transferred between the debugger signal interface controller and the debugger using Application Programming Interface (API) calls.

27.   (Original) A method as claimed in Claim 19, further comprising the step of:

writing at least one of the software routines into the memory via the debugger under the control of the debugger signal interface controller.

28.   (Original) A method as claimed in Claim 18, wherein multiple processing units are provided, and a corresponding multiple of debugger signal interface controllers are provided, each debugger signal interface controller communicating with the same debugger to cause their respective test actions to be performed.

29. (Original) A method as claimed in Claim 18, wherein the timing of the execution of said software routines is co-ordinated with the sequence of verification tests.

30. (Original) A method as claimed in Claim 18, wherein the system under verification comprises a plurality of components, each signal interface controller being associated with one of said components.

31. (Original) A method as claimed in Claim 30, wherein the processing unit forms one of the components of the system under verification.

32. (Original) A method as claimed in Claim 18, wherein the processing unit is provided externally to the system under verification.

33. (Original) A method as claimed in Claim 18, wherein the processing unit comprises a representation of a processor on which the software routines are intended to be executed.

34. (Original) A method as claimed in Claim 18, wherein the system under verification comprises a hardware simulator which in response to said one or more stimulus signals generates said one or more response signals simulating a response of a

data processing apparatus to said one or more stimulus signals if applied to said data processing apparatus.

35. (Previously Presented) A computer program product embodied on a computer-readable medium for use in performing a sequence of verification tests to perform hardware and software co-verification on a system under verification, the computer program product comprising:

a plurality of signal interface controller code blocks operable to be coupled to said system under verification, each signal interface controller code block, when executed, causes a computer to perform one or more test actions transferring at least one of one or more stimulus signals and one or more response signals between a corresponding portion of the system under verification and said signal interface controller code block during performance of said sequence of verification tests;

debugger code, when executed, causes the computer to control operation of a processing unit associated with the system under verification, the processing unit being operable to execute software routines;

a debugger signal interface controller code block, when executed, causes the computer to interface with the debugger code and to perform one or more test actions transferring at least one of one or more stimulus signals and one or more response signals between the debugger code and said debugger signal interface controller code block during performance of said sequence of verification tests; and

test manager code coupled to said plurality of signal interface controller

code blocks and the debugger signal interface controller code block and, when executed,

causes the computer to transfer test controlling messages to said plurality of signal

interface controller code blocks and the debugger signal interface controller code block

identifying the test actions to be performed;

the test manager code, when executed, causes the computer to control the

operation of the processing unit via the debugger signal interface controller code block

and the debugger code in order to co-ordinate the execution of said software routines with

the sequence of verification tests.

36.   (Previously Presented) A computer program product as claimed in Claim

35, wherein the software routines are stored in a memory, the debugger signal interface

controller code block being provided with an address indication identifying the addresses

of the software routines within the memory, upon receipt of a test controlling message

from the test manager code, the debugger signal interface controller code block, when

executed, causes the computer to generate a corresponding test action with reference to

the address indication.

37.   (Original) A computer program product as claimed in Claim 36, wherein

status data is stored in a status memory, the processing unit being operable to execute

monitoring code to monitor the status data in order to identify any changes to the status

data and to then execute at least one of said software routines as identified by the change in status data, the corresponding test action being arranged to cause updated status data identifying a corresponding one of said software routines to be stored in the status memory under the control of the debugger code.

38.   (Original) A computer program product as claimed in Claim 37, wherein the debugger code is operable to cause the processing unit to store the updated status data in the status memory, whereafter the processing unit reverts to executing the monitoring code.

39.   (Original) A computer program product as claimed in Claim 36, wherein the processing unit has a plurality of registers associated therewith, and the corresponding test action is arranged to cause data in a selected register of said plurality of registers to be updated under the control of the debugger code, such that the processing unit will then execute a corresponding one of said software routines.

40.   (Original) A computer program product as claimed in Claim 39, wherein the selected register is operable to store a program counter value and the corresponding test action is arranged to cause the program counter value to be updated in order to cause the processing unit to branch to the corresponding one of said software routines.

41.   (Previously Presented) A computer program product as claimed in Claim

35, wherein upon execution of one of said software routines, the processing unit is

operable to update status data indicative of whether the software routine completed

successfully, the debugger signal interface controller code block, when executed, causes

the computer to perform a predetermined test action in order to cause a breakpoint to be

set by the debugger code which is triggered when the processing unit performs said

update of the status data.

42.   (Previously Presented) A computer program product as claimed in Claim

41, wherein the debugger code, when executed, causes the computer to issue a callback

event to the debugger signal interface controller code block upon triggering of the

breakpoint.

43.   (Original) A computer program product as claimed in Claim 35, wherein

said stimulus and/or response signals are transferred between the debugger signal

interface controller code block and the debugger code using Application Programming

Interface (API) calls.

44.   (Previously Presented) A computer program product as claimed in Claim

36, wherein at least one of the software routines is written into the memory via the

debugger code under the control of the computer executing the debugger signal interface controller code block.

45.   (Previously Presented) A computer program product as claimed in Claim 35, wherein multiple processing units are provided, and a corresponding multiple of debugger signal interface controller code blocks are provided, each debugger signal interface controller code block, when executed, causes the computer to communicate with the same debugger code to cause their respective test actions to be performed.

46.   (Original) A computer program product as claimed in Claim 35, wherein the timing of the execution of said software routines is co-ordinated with the sequence of verification tests.

47.   (Original) A computer program product as claimed in Claim 35, wherein the system under verification comprises a plurality of components, each signal interface controller code block being associated with one of said components.

48.   (Original) A computer program product as claimed in Claim 47, wherein the processing unit forms one of the components of the system under verification.

49.   (Original) A computer program product as claimed in Claim 35, wherein the processing unit is provided externally to the system under verification.

50.   (Original) A computer program product as claimed in Claim 35, wherein the processing unit comprises a representation of a processor on which the software routines are intended to be executed.

51.   (Previously Presented) A computer program product as claimed in Claim 35, wherein the system under verification comprises hardware simulator code, when executed, causes the computer, in response to said one or more stimulus signals, to generate said one or more response signals simulating a response of a data processing apparatus to said one or more stimulus signals if applied to said data processing apparatus.

X. EVIDENCE APPENDIX

There is no evidence appendix.

XI. RELATED PROCEEDINGS APPENDIX

There is no related proceedings appendix.